Term Project Documentation (Tank Wars and Super Rainbow Reef)

CSC 413 – Summer 2018 Christopher Bryan Rosana

Tank Wars Repo:

https://github.com/chrisrosana/Tanks-movement

Super Rainbow Reef Repo:

https://github.com/chrisrosana/Rainbow-Reef-Game

Project Overview:

In this project, we implement two 2D games written in Java. For the first game, we implement the Tank Wars game. In Tank Wars, there are two players playing against each other, and the objective is to attack and destroy the player's opponent, which is the second player. For the second game, we get to choose a list of games that the professor provides to us. I choose the Super Rainbow Reef as my second game. This game is very similar to the Brick Breaker game which is to hit bricks and get the highest score each time you hit and destroy them, but with the addition of to destroy the octopus named Biglegs to proceed and finish the levels of the game. The goal of this project is to get a good practice of Object Oriented Programming and code reusability. In addition, the intended purpose of these two games, is to implement the first game – the tank game - and re-use a good portion of the first game to write the second game.

Development Environment:

In this assignment, my Java Development Kit version was on JDK version 8 and my IDE on IntelliJ IDEA Ultimate Edition on a macOS High Sierra.

To run the game in IntelliJ:

1. Download the project from:

   For the Tank Wars Game: https://github.com/chrisrosana/Tanks-movement

   For the Rainbow Reef Game: https://github.com/chrisrosana/Rainbow-Reef-Game

2. At this time, you downloaded a zip file, unzip it to any directory you prefer.
3. Open IntelliJ and select “File from the dropdown menu. Hover to “New” then select “Project from Existing Sources…”
4. Navigate the project folder and click “Open”
5. Select “Create project from existing sources” then click “Next”.
6. You can choose to rename the project name, edit the location and edit the format, but if you want to remain it the same, click “Next”.
7. Click “Next” on the bottom right.
8. Click “Next” on the bottom right again.
9. Click “Next” on the bottom right again.
10. Choose the SDK to run the project, then click “Next” on the bottom right again.
11. Click “Finish”.
12. On the left section, right click on TankApplication.java or Application.java and click “Run ‘TankApplication.main()’” or “Run ‘Application.main()’”

Tank Wars Game

Controls and Objectives:

The objective of this game is to play against with the opponent which is the other player. It includes two players, which they fight one another by firing bullets to the opponent and look for some power ups to increase your chances and opportunities to win the game.

For player 1, the controls are:

A key = rotate left

W key = move forwards

S key = move backwards

D key = rotate right

SPACE key = fire a bullet

For player 2, the controls are:

LEFT arrow key = rotate left

UP arrow key = move forwards

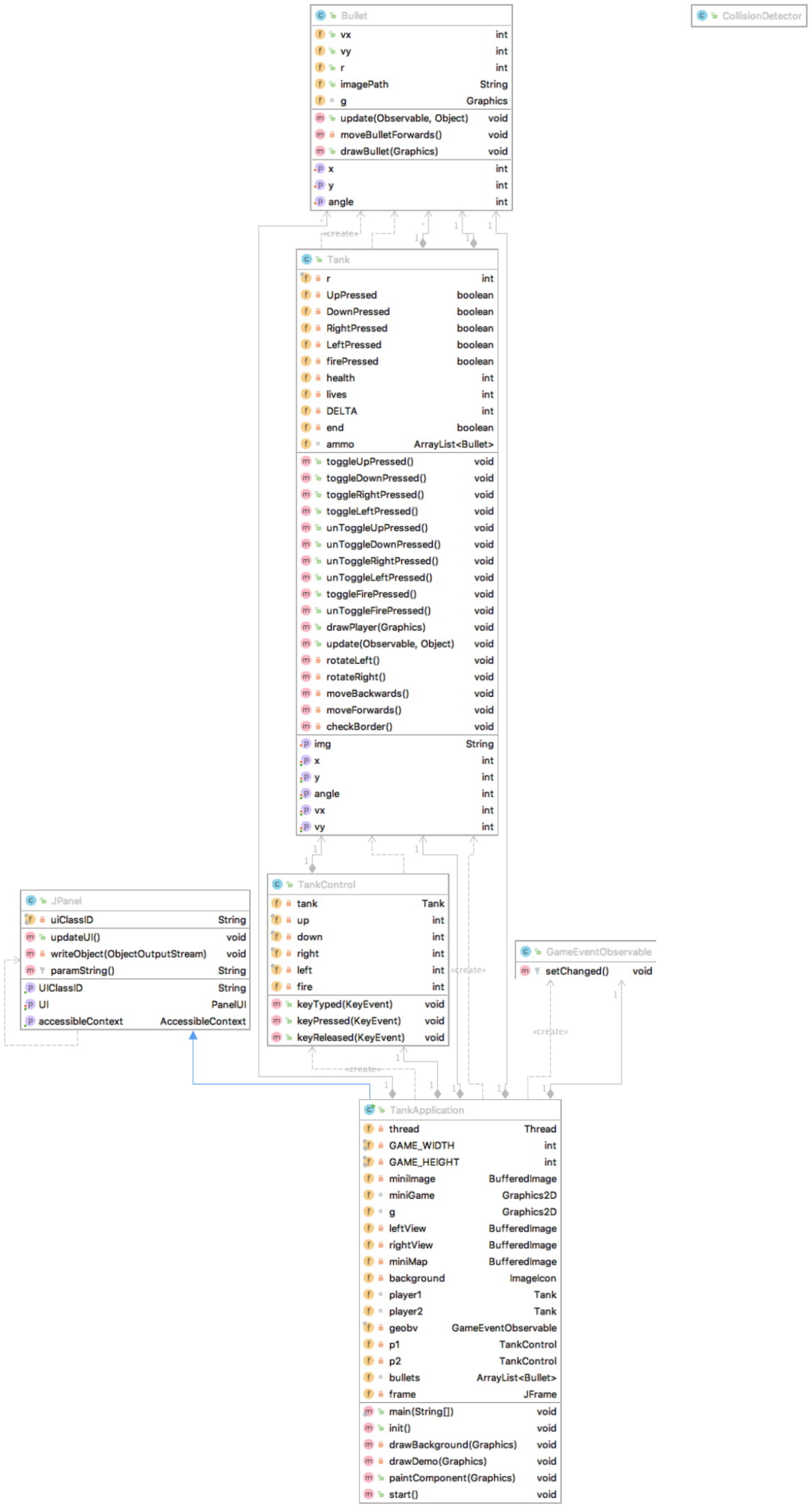
DOWN arrow key = move backwards

RIGHT arrow key = rotate right

ENTER key = fire a bullet

Technical Overview and Project Components:

Tank Wars Implementation



For my Tank Wars project, it is incomplete, and it includes a few classes:

1. TankApplication.java
   - This class holds the main method to start the game. It extends to JPanel and it holds the concept methods to set up the window, the thread of the game, the initialization of the game – loading the image resources, set the controls and the view of the game, paint components to draw and display the images, set up the split screen for the players, and objects to add on the application of the game, and updates the game each time any object like the tank moves.
2. Tank.java
   - This class is used to draw the tank and set the x and y coordinates to display the tank in the TankApplication.java. It also sets up the control of the tank such as moving backwards and forwards, rotate left and right, and firing the bullet and update its location. In addition, it also draws the health bar and the lives of the tank.
3. TankControl.java
   - This mainly sets up the control of the tank when the key from the keyboard is pressed and released and updates its current condition.
4. GameEventObservable.java
   - This basically extends from observable class and watches and sets any changes that is happening on the tank like when it moves and fires the bullets.
5. Bullet.java

- This is similar to the Tank.java but a dumb version of it. This is where the bullets are drawn and moves the bullet independently in one linear direction.

6. CollisionDetector.java
   - This file is empty and it has no classes in it. It is incomplete and it's supposed to detect any objects such as the two tanks when it collides or when a tank and wall collides.

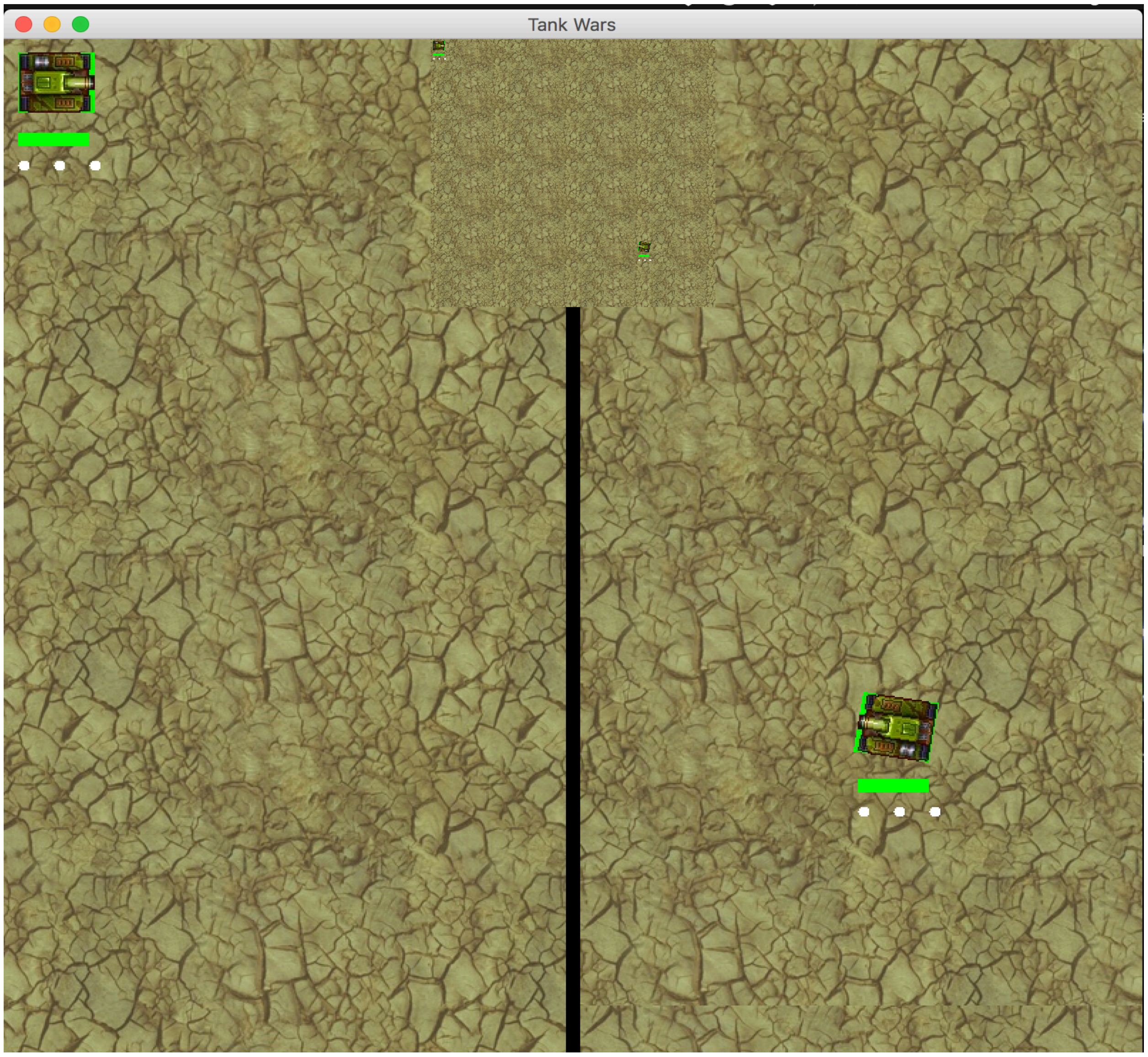
Assumptions, Discussions, and Reflections:

As I said earlier, this project is incomplete, but it still meets some of the requirements of the game. It has two players, and each player have each tank and it moves forwards and backwards using up and down arrow keys for player 2, and W and S keys for player 1. Each tank rotates, so they can move in all directions. The game has a split screen for the players to view their tanks and it has the mini-map to see the birds eye view and the overall of the game. Each tank has its own health bars and lives though it does count the lives for each tank. Some of the incomplete requirements are: Each tank doesn't shoot bullets, doesn't have unbreakable and breakable walls each time a bullet hits it, and tanks and bullets does collide with walls and tanks.

For designing of the game, I used our professor's tank rotation example to know how the tank moves and rotates. However, his classes differ from mines. It extends to JFrame which means it only accepts one component and to add a tank it needs a JPanel to add components in a container, so I did it in the other way where I extend my class to JPanel to add more components in my JFrame. I also collaborate with some of my study groups Carina, Derick, Diana and Gabriel to help on figuring on how to extend all my components to JPanel.

In this project, I found many challenges. I found it a very time consuming. I struggled on the bullets and have a hard time figuring when I pressed the fire key it supposed to draw the bullets. And that's where I found my weakness in observable class, observer class, and update methods. In addition, I find that I spend a lot of time on designing or the look and creativity of my game, and doing my split screen. This cause to ran out of my time to do all the other requirements.

Screenshot of my Tank Wars Game



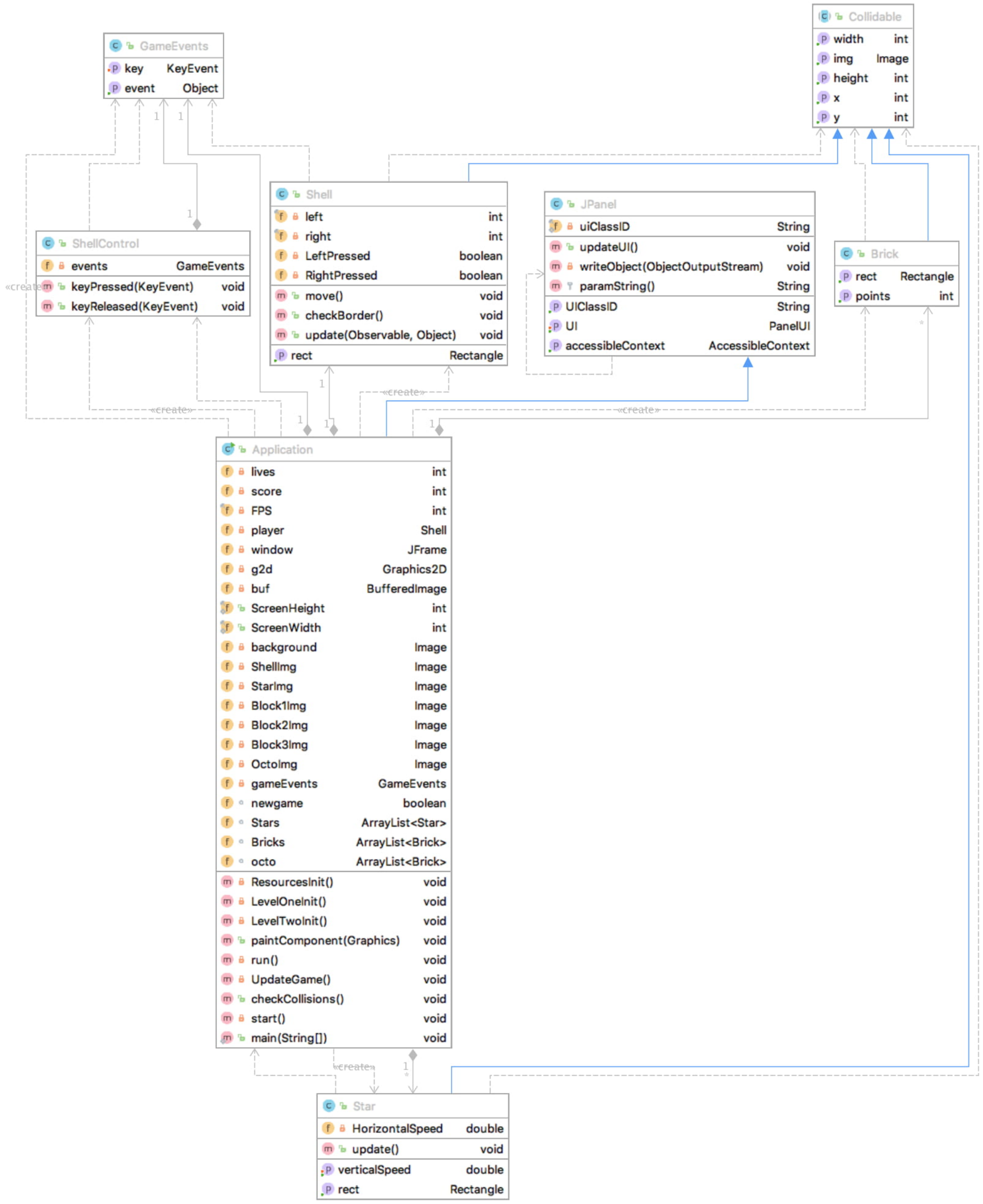
Super Rainbow Reef Game

Controls and Objectives:

The goal of the game is to hit many bricks as you can to get the highest score and defeat Biglegs – an octopus, to get to the next level and finish the game. The game is only for one player and the controls are the LEFT and RIGHT arrow keys to move left and right respectively.

Technical Overview and Project Components:

Super Rainbow Reef Implementation



For my Rainbow Reef project. It supposed to reuse some of my code or classes in Tank Wars project. It includes these classes:

1. Application.java
   - Similar to my Tank Wars, this class holds the main method to start the game. It extends to JPanel and it holds the concept methods to set the window, the thread of the game, the initialization of the game – loading the image resources, set the controls and the view of the game, paint components to draw and display the images, objects to add on the application of the game, and updates the game each time any objects move on the screen. However, I added my collision detection method here which is called checkcollisions() to detect any collision so the star can bounce on any objects and updates any objects happening when the star hits a brick or the shell like counting the points and lives.
2. Star.java
   - Quite similar to Bullet class in Tank Wars though it extends to Collidable class to check collisions. It is where the stars are drawn and moves independently in any direction, and bounces to any bricks and borders each it hits them. I also added a gravity of the Star so when it bounce it acts natural as it falls from the earth. It has getRect() method to invisibly draw a rectangle to use it for analyzing any collisions to any objects on the screen.
3. Shell.java
   - Like the Tank class in Tank Wars though it extends to Collidable class like the Star class. this class is used to draw the shell and set the coordinates to display the

shell in the Application.java. It also sets up the control of the shell such as moving left and right and update its location. In addition, it also has getRect() method just like the Star.java to analyze any collisions.

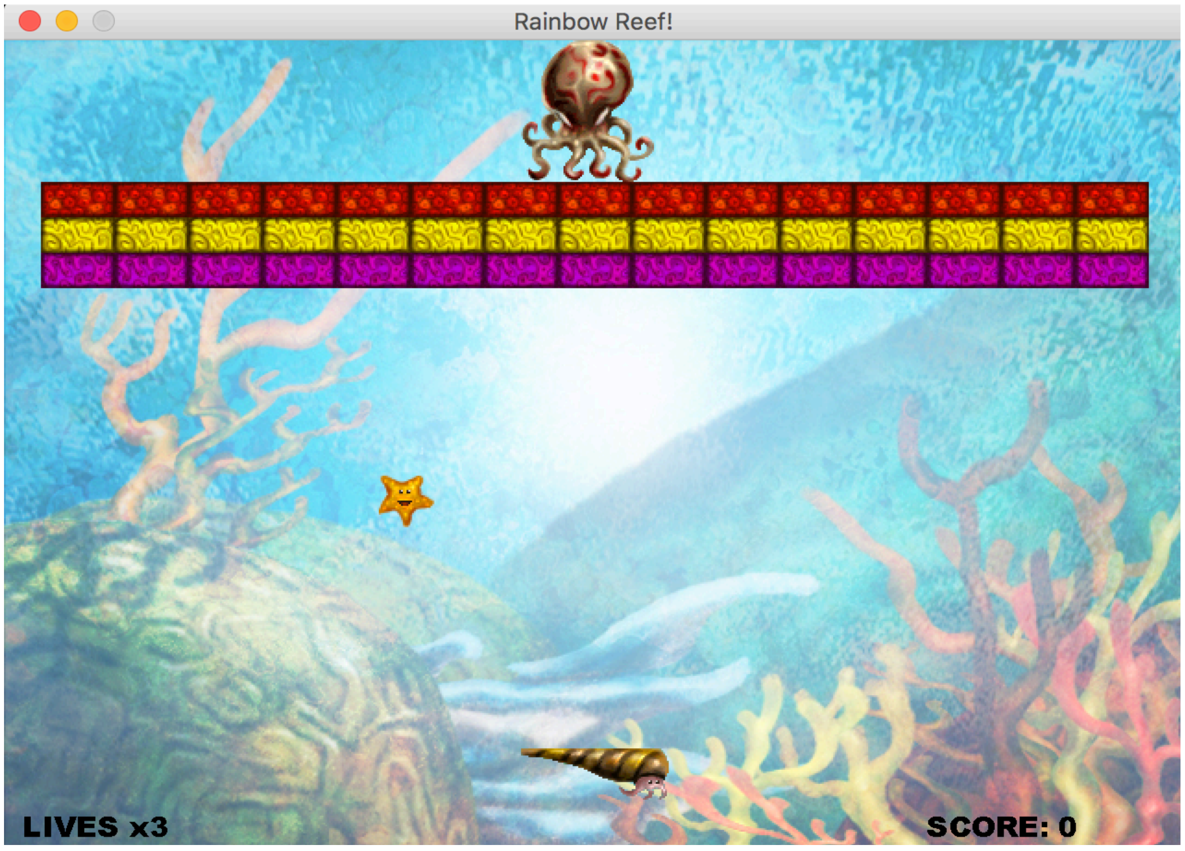
4. ShellControl.java
   - This mainly sets up the control of the shell like the TankControl class when the key from the keyboard is pressed and released.
5. Brick.java
   - This class extends to Collidable just the same as Shell and Star, and it sets the coordinates to place the bricks on the screen and gets points each time the star hits it.
6. Collidable.java
   - This is an abstract class and it sets the images and coordinates for each class that extends to this. It also gets the images, and the coordinates and widths for any collisions and displays.
7. GameEvents.java
   - This implements Observable class that will notify the observer which observes when some events happen like each time a key from a keyboard is pressed and released.

Assumptions, Discussions, and Reflections:

I found this project more of a relief when building and designing it. I learned a lot of building this game by chatting with Gabriel and other of my study group in class on how to bounce my star and move my shell and most importantly watching videos online and how I can implement what I learn into this games. A video that I watched on how I can do the collision detection helps me on how I can do my collisions on this game which I found it difficult when I did my tank game. Some obstacles I found are when star hits a brick or shell: how I can remove the brick when it hits the star and how I can bounce the star in a different direction each time it hits the wall, shell, or brick. I found that I should do an arraylist of bricks to remove and add them. For the BigLegs I also did that as an arraylist but as soon as it hits the last Biglegs, it moves on to the next level where the player faces the two BigLegs which the player needs to defeat.

I think I completed the requirements of the game such as the collisions, controls, counting the scores and lives and the flow of the game.

Screenshot of my Rainbow Reef Game:


Rainbow Reef!
LIVES x3
SCORE: 0